

# Machine Learning Evaluation & Supervised Learning

Model Evaluation



# Rezki Trianto

Data guy who spent the last 7 years to work around data analytics and machine learning. Working in a unicorn company with various domain expertise to work with, from an e-commerce, OTA, and ride-hailing company.



Rezki Trianto

[linkedin.com/in/rezkitrianto](https://www.linkedin.com/in/rezkitrianto)

## Education Background



**2010-2014**  
Bachelor Degree  
Computer Science



**2015-2017**  
Master Degree  
Computer Science

### Rezki Trianto

5+ years experience in Data Science & Analytics

# Hands On Required


**Hands - On : 1. Model Evaluation**

Klik disini untuk  
mengakses folder Hands  
On

# Topik Machine Learning Model Evaluation

- ☐ Pengenalan evaluasi model
- ☐ Evaluasi model pada Regresi (Part. 1)
- ☐ Evaluasi model pada Regresi (Part. 2)
- ☐ Implementasi Python - Eval Regresi
- ☐ Evaluasi model pada Klasifikasi (Part. 1)
- ☐ Evaluasi model pada Klasifikasi (Part. 2)
- ☐ Implementasi Python - Eval Klasifikasi
- ☐ Bias-Variance Tradeoff - Overfit & Underfit
- ☐ Bias-Variance Tradeoff - How to solve
- ☐ Model Validation

# Topik Machine Learning Model Evaluation

 ☐ Pengenalan evaluasi model

☐ Evaluasi model pada Regresi (Part. 1)

☐ Evaluasi model pada Regresi (Part. 2)

☐ Implementasi Python - Eval Regresi

☐ Evaluasi model pada Klasifikasi (Part. 1)

☐ Evaluasi model pada Klasifikasi (Part. 2)

☐ Implementasi Python - Eval Klasifikasi

☐ Bias-Variance Tradeoff - Overfit & Underfit

☐ Bias-Variance Tradeoff - How to solve

☐ Model Validation



# Mengapa sebuah model perlu di evaluasi?

Dengan melakukan evaluasi, kualitas dari model akan terjaga, sehingga meminimalisir kesalahan dari hasil prediksi.

Contoh use case: **Propensity model in marketing**

- **Kasus False Positive**

(Hasil prediksi seorang customer akan melakukan transaksi, sedangkan data aktualnya tidak)

Jika customer ini diberikan marketing action, hanya akan memperbesar marketing cost, tanpa hasil apapun

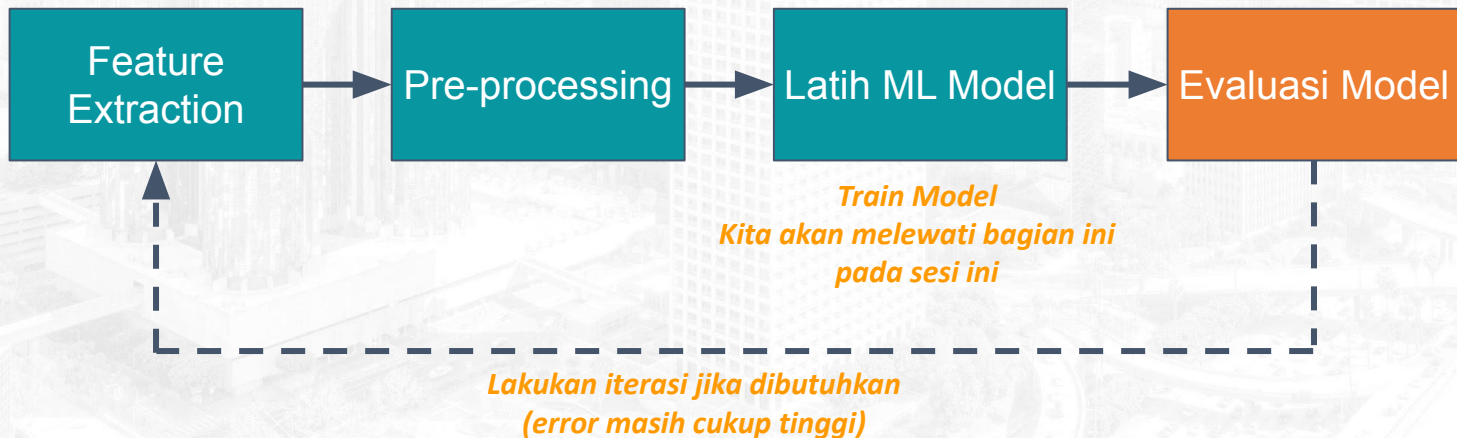
- **Kasus False Negative**

(Hasil prediksi seorang customer tidak melakukan transaksi, sedangkan data aktualnya akan ber-transaksi)

Jika customer ini tidak diberikan marketing action, akan ada potensi transaction / revenue loss

# Mengapa Sebuah Model Perlu di evaluasi?

Dengan melakukan evaluasi, kualitas dari model akan terjaga, sehingga meminimalisir kesalahan dari hasil prediksi.



## Jenis-jenis task machine learning

### Supervised Learning

- Tersedia data + target
- Klasifikasi, regresi

### Unsupervised Learning

- Tersedia data tanpa target
- Clustering, representation learning

### Reinforcement Learning

- Data tidak tersedia (???), hanya aturan
- Melatih 'agen' dalam suatu 'task'

*\* diluar scope Bootcamp*

**Setiap task Machine Learning memiliki pemodelan evaluasi yang berbeda-beda**



# Topik Machine Learning Model Evaluation



Pengenalan evaluasi model



Evaluasi model pada Regresi (Part. 1)



Evaluasi model pada Regresi (Part. 2)



Implementasi Python - Eval Regresi



Evaluasi model pada Klasifikasi (Part. 1)



Evaluasi model pada Klasifikasi (Part. 2)



Implementasi Python - Eval Klasifikasi



Bias-Variance Tradeoff - Overfit & Underfit



Bias-Variance Tradeoff - How to solve



Model Validation

# Topik Machine Learning Model Evaluation



Pengenalan evaluasi model



Evaluasi model pada Regresi (Part. 1)



Evaluasi model pada Regresi (Part. 2)



Implementasi Python - Eval Regresi



Evaluasi model pada Klasifikasi (Part. 1)



Evaluasi model pada Klasifikasi (Part. 2)



Implementasi Python - Eval Klasifikasi



Bias-Variance Tradeoff - Overfit & Underfit

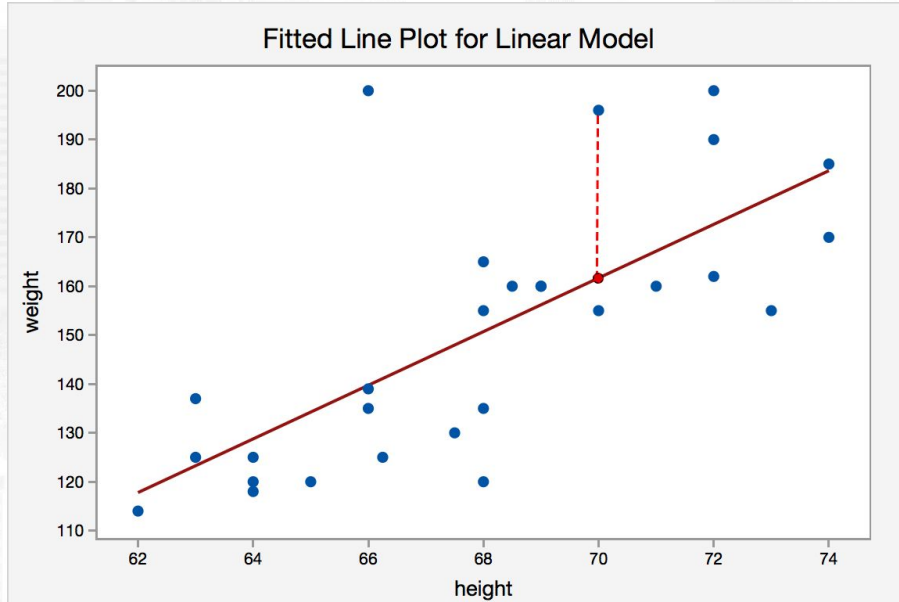


Bias-Variance Tradeoff - How to solve



Model Validation

# Evaluasi Model: Regresi

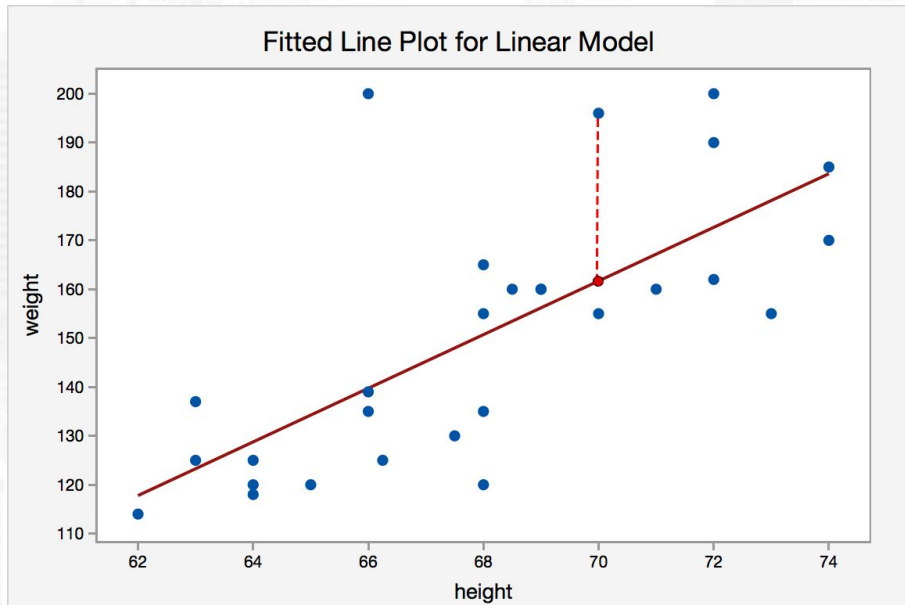


Kasus Regresi dilakukan untuk:

1. Menemukan garis linear yang optimal pada setiap data point
2. Melakukan prediksi nilai Y berdasarkan nilai x

(Detail akan dibahas pada modul Regresi)

# Evaluasi Model: Regresi



Evaluasi yang biasa digunakan adalah dengan menghitung jarak antara hasil prediksi dengan posisi asalnya (error)

1. RMSE (root mean square error)
2. MAE (mean absolute error)

$$\text{RMSE} = \sqrt{\frac{1}{n} \sum_{j=1}^n (y_j - \hat{y}_j)^2}$$

$$\text{MAE} = \frac{1}{n} \sum_{j=1}^n |y_j - \hat{y}_j|$$

Error yang lebih kecil lebih baik.

# Kapan menggunakan MAE/RMSE?

CASE 1: Evenly distributed errors

ID	Error	Error	Error^2
1	2	2	4
2	2	2	4
3	2	2	4
4	2	2	4
5	2	2	4
6	2	2	4
7	2	2	4
8	2	2	4
9	2	2	4
10	2	2	4

MAE	RMSE
2.000	2.000

CASE 2: Small variance in errors

ID	Error	Error	Error^2
1	1	1	1
2	1	1	1
3	1	1	1
4	1	1	1
5	1	1	1
6	3	3	9
7	3	3	9
8	3	3	9
9	3	3	9
10	3	3	9

MAE	RMSE
2.000	2.236

CASE 3: Large error outlier

ID	Error	Error	Error^2
1	0	0	0
2	0	0	0
3	0	0	0
4	0	0	0
5	0	0	0
6	0	0	0
7	0	0	0
8	0	0	0
9	0	0	0
10	20	20	400

MAE	RMSE
2.000	6.325

RMSE mempunyai keuntungan dengan memberikan error yang besar jika terdapat outlier, sehingga menghasilkan pengukuran yang tepat untuk beberapa kasus yang lebih sensitif





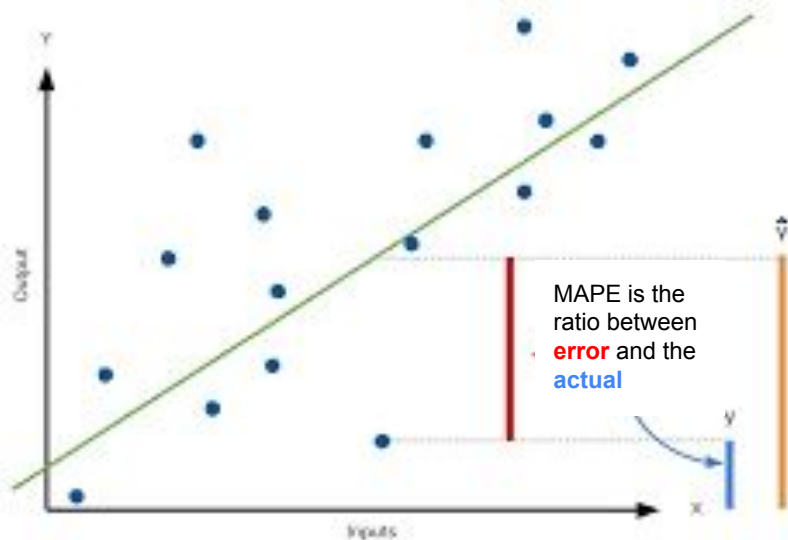
# Topik Machine Learning Model Evaluation

- ☒ Pengenalan evaluasi model
- ☒ Evaluasi model pada Regresi (Part. 1)
- ☐ Evaluasi model pada Regresi (Part. 2)
- ☐ Implementasi Python - Eval Regresi
- ☐ Evaluasi model pada Klasifikasi (Part. 1)
- ☐ Evaluasi model pada Klasifikasi (Part. 2)
- ☐ Implementasi Python - Eval Klasifikasi
- ☐ Bias-Variance Tradeoff - Overfit & Underfit
- ☐ Bias-Variance Tradeoff - How to solve
- ☐ Model Validation

# Topik Machine Learning Model Evaluation

- ☒ Pengenalan evaluasi model
- ☒ Evaluasi model pada Regresi (Part. 1)
- ☒ Evaluasi model pada Regresi (Part. 2)
- ☐ Implementasi Python - Eval Regresi
- ☐ Evaluasi model pada Klasifikasi (Part. 1)
- ☐ Evaluasi model pada Klasifikasi (Part. 2)
- ☐ Implementasi Python - Eval Klasifikasi
- ☐ Bias-Variance Tradeoff - Overfit & Underfit
- ☐ Bias-Variance Tradeoff - How to solve
- ☐ Model Validation

# Evaluasi Model: Regresi - MAPE



**MAPE (Mean Absolute Percentage Error)** mengukur besarnya error dalam satuan persen dan metrics ini lebih mudah di interpretasi terutama jika pengukuran error telah dilakukan transformasi feature.

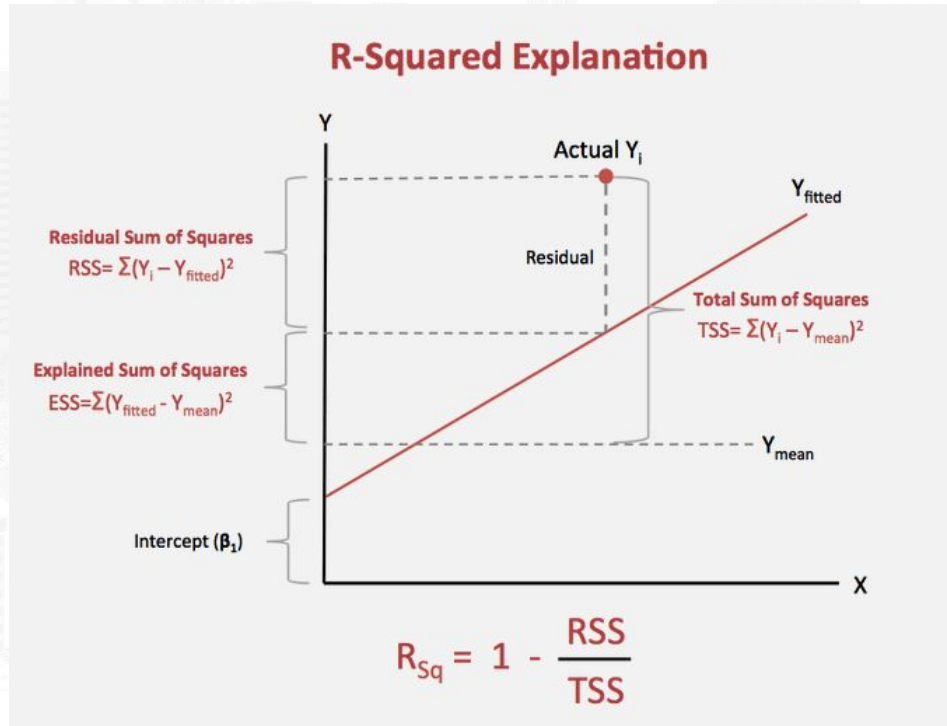
Mengambil rasio antara jarak (error) dengan  $y_{\text{actual}}$ .

MAPE	Interpretation
<10	Highly accurate forecasting
10-20	Good forecasting
20-50	Reasonable forecasting
>50	Inaccurate forecasting

Persentase yang lebih kecil lebih baik.

Source: Lewis (1982, p. 40)

# Evaluasi Model: Regresi - $R^2$



$R^2$  mengukur keberhasilan model regresi yang digunakan.

Nilainya berkisar antara 0-1, mengindikasikan seberapa besar variabel independen mempengaruhi variabel dependen.

Semakin mendekati angka 1, model semakin baik. Dan umumnya, semakin besar nilai  $r^2$  akan mempengaruhi error yang semakin kecil.


Score yang lebih besar lebih baik.



# Topik Machine Learning Model Evaluation

- ☒ Pengenalan evaluasi model
- ☒ Evaluasi model pada Regresi (Part. 1)
- ☒ Evaluasi model pada Regresi (Part. 2)
- ☐ Implementasi Python - Eval Regresi
- ☐ Evaluasi model pada Klasifikasi (Part. 1)
- ☐ Evaluasi model pada Klasifikasi (Part. 2)
- ☐ Implementasi Python - Eval Klasifikasi
- ☐ Bias-Variance Tradeoff (Part 1)
- ☐ Bias-Variance Tradeoff (Part 2)
- ☐ Model Validation

# Topik Machine Learning Model Evaluation

- ☒ Pengenalan evaluasi model
- ☒ Evaluasi model pada Regresi (Part. 1)
- ☒ Evaluasi model pada Regresi (Part. 2)
- ☒  Implementasi Python - Eval Regresi
- ☐ Evaluasi model pada Klasifikasi (Part. 1)
- ☐ Evaluasi model pada Klasifikasi (Part. 2)
- ☐ Implementasi Python - Eval Klasifikasi
- ☐ Bias-Variance Tradeoff - Overfit & Underfit
- ☐ Bias-Variance Tradeoff - How to solve
- ☐ Model Validation

# Contoh Implementasi di Python

```
y_true = [3, -0.5, 2, 7] # actual data
y_pred = [2.5, 0.0, 2, 8] # anggap ini hasil prediksi dari model
```

```
# MAE
from sklearn.metrics import mean_absolute_error
mean_absolute_error(y_true, y_pred) # hitung MAE
```

Import package dari sklearn

$$\begin{aligned}
 \text{MAE} &= \frac{(|3-2.5| + |-0.5-0| + |2-2| + |7-8|)}{4} \\
 &= \frac{(0.5 + 0.5 + 0 + 1)}{4} \\
 &= 0.5
 \end{aligned}$$

Output:

```
1 from sklearn.metrics import mean_absolute_error
2 mean_absolute_error(y_true, y_pred)

executed in 570ms, finished 23:50:12 2021-03-15

0.5
```

# Contoh Implementasi di Python

```
y_true = [3, -0.5, 2, 7] # actual data
y_pred = [2.5, 0.0, 2, 8] # anggap ini hasil prediksi dari model
```

```
# RMSE
from sklearn.metrics import mean_squared_error
mean_squared_error(y_true, y_pred, squared=False) # hitung RMSE
```

$$\begin{aligned}
 \text{RMSE} &= \sqrt{\frac{(3-2.5)^2 + (-0.5-0)^2 + (2-2)^2 + (7-8)^2}{4}} \\
 &= \sqrt{\frac{0.25 + 0.25 + 0 + 1}{4}} \\
 &= 0.61
 \end{aligned}$$

## Output:

```
1 from sklearn.metrics import mean_squared_error
2 mean_squared_error(y_true, y_pred, squared=False)
```

executed in 8ms, finished 00:03:01 2021-03-16

0.6123724356957945

# Contoh Implementasi di Python

```
# R2
from sklearn.metrics import r2_score
r2_score(y_true, y_pred) # hitung R2 score
```

```
# MAPE
from sklearn.metrics import mean_absolute_percentage_error
mean_absolute_percentage_error(y_true, y_pred) # hitung MAPE
```



## Challenge Time 1 (3 mins)

```
actual = [4, 6, -9, 8, 10]  
predicted = [1, 2, -8, 0, 10]
```

Berapa nilai MAE dan RMSE dari data actual dan predicted diatas?  
Gunakan script python untuk menghitungnya

## Challenge Time 1 (3 mins)

```
actual = [4, 6, -9, 8, 10]  
predicted = [1, 2, -8, 0, 10]
```

Berapa nilai MAE dan RMSE dari data actual dan predicted diatas?  
Gunakan script python untuk menghitungnya

MAE: 3.2

RMSE: 4.2

# Topik Machine Learning Model Evaluation

- ☒ Pengenalan evaluasi model
- ☒ Evaluasi model pada Regresi (Part. 1)
- ☒ Evaluasi model pada Regresi (Part. 2)
- ☒ Implementasi Python - Eval Regresi
- ☐ Evaluasi model pada Klasifikasi (Part. 1)
- ☐ Evaluasi model pada Klasifikasi (Part. 2)
- ☐ Implementasi Python - Eval Klasifikasi
- ☐ Bias-Variance Tradeoff - Overfit & Underfit
- ☐ Bias-Variance Tradeoff - How to solve
- ☐ Model Validation

# Topik Machine Learning Model Evaluation

- ☒ Pengenalan evaluasi model
- ☒ Evaluasi model pada Regresi (Part. 1)
- ☒ Evaluasi model pada Regresi (Part. 2)
- ☒ Implementasi Python - Eval Regresi
- ☒ Evaluasi model pada Klasifikasi (Part. 1)
- ☐ Evaluasi model pada Klasifikasi (Part. 2)
- ☐ Implementasi Python - Eval Klasifikasi
- ☐ Bias-Variance Tradeoff - Overfit & Underfit
- ☐ Bias-Variance Tradeoff - How to solve
- ☐ Model Validation

# Evaluasi Model Klasifikasi

## Classification Result - aka. Confusion Matrix

	Predicted class POSITIVE (spam 📧 )	Predicted class NEGATIVE (normal 📧 )
Actual class POSITIVE (spam 📧 )	TRUE POSITIVE (TP) 📧 📧 320	FALSE NEGATIVE (FN) 📧 📧 43
Actual class NEGATIVE (normal 📧 )	FALSE POSITIVE (FP) 📧 📧 20	TRUE NEGATIVE (TN) 📧 📧 538

### Contoh Kasus

Prediksi apakah sebuah email merupakan **spam** atau normal.

- **True Positive & True Negative:**  
Hasil Prediksi yang diinginkan (tepat)
- **False Positive & False Negative:**  
Hasil prediksi yang tidak tepat dan tidak diinginkan



# Accuracy

	Predicted class POSITIVE (spam 📧 )	Predicted class NEGATIVE (normal 📧 )
Actual class POSITIVE (spam 📧 )	TRUE POSITIVE (TP) 📧 📧 320	FALSE NEGATIVE (FN) 📧 📧 43
Actual class NEGATIVE (normal 📧 )	FALSE POSITIVE (FP) 📧 📧 20	TRUE NEGATIVE (TN) 📧 📧 538

$$\text{Accuracy} = \frac{\text{True Positive} + \text{True Negative}}{\text{Total}}$$

Metrics Akurasi biasa digunakan ketika masing-masing label mempunyai kepentingan yang sama dan jumlah labelnya seimbang.

Jika tidak seimbang, maka hasil prediksi akan cenderung positif atau negatif (sesuai dengan jumlah data yang dominan), dan mengakibatkan perhitungan tidak adil dan kurang valid.

Contoh : prediksi gender

$$\begin{aligned} \text{Accuracy} &= (320+538)/(320+538+43+20) \\ &= 0.93 \end{aligned}$$





Score accuracy yang lebih besar lebih baik.

## Apakah mengukur **akurasi** saja dari sebuah model cukup dalam melakukan evaluasi?

Tidak, kita harus menyesuaikan masing-masing kasusnya.

Akurasi biasanya cocok digunakan pada saat perbandingan jumlah label data relatif sama, dan kepentingan antar labelnya juga sama.

# Precision

	Predicted class POSITIVE (spam 📧 )	Predicted class NEGATIVE (normal 📧 )
Actual class POSITIVE (spam 📧 )	<b>TRUE POSITIVE (TP)</b>  <div>320</div>	<b>FALSE NEGATIVE (FN)</b>  <div>43</div>
Actual class NEGATIVE (normal 📧 )	<b>FALSE POSITIVE (FP)</b>  <div>20</div>	<b>TRUE NEGATIVE (TN)</b>  <div>538</div>

$$\text{Precision} = \frac{\text{True Positive}}{\text{Actual Results}} \quad \text{or} \quad \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}}$$

$$\text{Precision} = (320)/(320+20) = 0.91$$

Biasa digunakan ketika kita lebih memperhatikan jumlah **False Positive (FP)** yang sebaiknya lebih sedikit dengan label yang seimbang.

Contoh kasus: deteksi spam. Mengurangi FP lebih cocok agar tidak ada email penting yang terlewat dan mengakibatkan experience yang buruk. Email spam masih dapat difilter manual oleh pengguna.

# Recall (True Positive Rate)

	Predicted class <b>POSITIVE</b> (spam 📧 )	Predicted class <b>NEGATIVE</b> (normal 📧 )
Actual class <b>POSITIVE</b> (spam 📧 )	<b>TRUE POSITIVE (TP)</b> 📧 📧 <div>320</div>	<b>FALSE NEGATIVE (FN)</b> 📧 📧 <div>43</div>
Actual class <b>NEGATIVE</b> (normal 📧 )	<b>FALSE POSITIVE (FP)</b> 📧 📧 <div>20</div>	<b>TRUE NEGATIVE (TN)</b> 📧 📧 <div>538</div>

$$\text{Recall} = \frac{\text{True Positive}}{\text{Predicted Results}} \quad \text{or} \quad \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}}$$

$$\text{Recall} = (320)/(320+43) = 0.88$$

Jika kita **tidak memperbolehkan nilai False Negative yang besar dan labelnya seimbang.**

Contoh kasus: Deteksi penyakit kanker  
 Prediksi penyakit kanker sangat sangat memperhatikan angka FN yang kecil karena hasilnya sangat sensitif terhadap dunia medis.

# Accuracy Paradox

## Cancer Prediction

Total Observation: 300

## Actual Class

Positive: 70

Negative: 230

**Goal:** membuat model machine learning untuk melakukan prediksi dari penyakit cancer



# Accuracy Paradox

**Model 1: Accuracy = 76%; Recall = 0%; Precision = 0%**

		Predicted	
		Positive	Negative
Actual	Positive	0 (TP)	70 (FN)
	Negative	0 (FP)	230 (TN)

# Accuracy Paradox

**Model 1: Accuracy = 76%; Recall = 0%; Precision = 0%**

		Predicted	
		Positive	Negative
Actual	Positive	0 (TP)	70 (FN)
	Negative	0 (FP)	230 (TN)

**Model 2: Accuracy = 73%; Recall = 14%; Precision = 33.3%**

		Predicted	
		Positive	Negative
Actual	Positive	10 (TP)	60 (FN)
	Negative	20 (FP)	210 (TN)

# Topik Machine Learning Model Evaluation

- ☒ Pengenalan evaluasi model
- ☒ Evaluasi model pada Regresi (Part. 1)
- ☒ Evaluasi model pada Regresi (Part. 2)
- ☒ Implementasi Python - Eval Regresi
- ☒ Evaluasi model pada Klasifikasi (Part. 1)
- ☐ Evaluasi model pada Klasifikasi (Part. 2)
- ☐ Implementasi Python - Eval Klasifikasi
- ☐ Bias-Variance Tradeoff - Overfit & Underfit
- ☐ Bias-Variance Tradeoff - How to solve
- ☐ Model Validation

# Topik Machine Learning Model Evaluation



Pengenalan evaluasi model



Evaluasi model pada Regresi (Part. 1)



Evaluasi model pada Regresi (Part. 2)



Implementasi Python - Eval Regresi



Evaluasi model pada Klasifikasi (Part. 1)



Evaluasi model pada Klasifikasi (Part. 2)



Implementasi Python - Eval Klasifikasi



Bias-Variance Tradeoff - Overfit & Underfit



Bias-Variance Tradeoff - How to solve



Model Validation

Di sesi sebelumnya:

## Prediksi label yang imbalance?

Algoritma pembelajaran pada Machine Learning biasanya bertujuan untuk memaksimalkan ukuran akurasi.

*Class imbalance* bermasalah karena kondisi ini membuat algoritma machine learning menjadi bodoh.

Bagaimana cara mengatasi *class imbalance*?

- Berikan ukuran akurasi yang lebih 'pintar' (sesi selanjutnya)
  - Hilangkan *class imbalance* pada data dengan over/undersampling



# Area Under ROC Curve (AUC) for Imbalance Case

## Drawing ROC Curve

<b>True Label</b>	0	0	1	1	0	0	1
<b>Pred Proba</b>	0.1	0.2	0.2	0.3	0.4	0.7	0.7

ROC curve melakukan plot terhadap True Positive Rate (TPR) dan False Positive Rate (FPR).

$$\text{TPR} = \text{TP} / (\text{TP} + \text{FN})$$

$$\text{FPR} = \text{FP} / (\text{FP} + \text{TN})$$

Area Under the Curve (AUC) sendiri mengukur kemampuan classifier untuk membedakan masing-masing class hasil prediksi.

# Area Under ROC Curve (AUC) for Imbalance Case

## Drawing ROC Curve

<b>True Label</b>	0	0	1	1	0	0	1
<b>Pred Proba</b>	0.1	0.2	0.2	0.3	0.4	0.7	0.7

threshold	TPR	FPR
<b>0.15</b>	$3/3 = 1$	$3/4 = 0.75$

ROC curve melakukan plot terhadap True Positive Rate (TPR) dan False Positive Rate (FPR).

$$\text{TPR} = \text{TP} / (\text{TP} + \text{FN})$$

$$\text{FPR} = \text{FP} / (\text{FP} + \text{TN})$$

Area Under the Curve (AUC) sendiri mengukur kemampuan classifier untuk membedakan masing-masing class hasil prediksi.

# Area Under ROC Curve (AUC) for Imbalance Case

## Drawing ROC Curve

<b>True Label</b>	0	0	1	1	0	0	1
<b>Pred Proba</b>	0.1	0.2	0.2	0.3	0.4	0.7	0.7

threshold	TPR	FPR
<b>0.15</b>	$3/3 = 1$	$3/4 = 0.75$
<b>0.25</b>	$2/3 = 0.67$	$2/4 = 0.5$

ROC curve melakukan plot terhadap True Positive Rate (TPR) dan False Positive Rate (FPR).

$$\text{TPR} = \text{TP} / (\text{TP} + \text{FN})$$

$$\text{FPR} = \text{FP} / (\text{FP} + \text{TN})$$

Area Under the Curve (AUC) sendiri mengukur kemampuan classifier untuk membedakan masing-masing class hasil prediksi.

# Area Under ROC Curve (AUC) for Imbalance Case

## Drawing ROC Curve

<b>True Label</b>	0	0	1	1	0	0	1
<b>Pred Proba</b>	0.1	0.2	0.2	0.3	0.4	0.7	0.7

threshold	TPR	FPR
<b>0.15</b>	$3/3 = 1$	$3/4 = 0.75$
<b>0.25</b>	$2/3 = 0.67$	$2/4 = 0.5$
<b>0.35</b>	$1/3 = 0.33$	$2/4 = 0.5$

ROC curve melakukan plot terhadap True Positive Rate (TPR) dan False Positive Rate (FPR).

$$\text{TPR} = \text{TP} / (\text{TP} + \text{FN})$$

$$\text{FPR} = \text{FP} / (\text{FP} + \text{TN})$$

Area Under the Curve (AUC) sendiri mengukur kemampuan classifier untuk membedakan masing-masing class hasil prediksi.

# Area Under ROC Curve (AUC) for Imbalance Case

## Drawing ROC Curve

<b>True Label</b>	0	0	1	1	0	0	1
<b>Pred Proba</b>	0.1	0.2	0.2	0.3	0.4	0.7	0.7

ROC curve melakukan plot terhadap True Positive Rate (TPR) dan False Positive Rate (FPR).

$$\text{TPR} = \text{TP} / (\text{TP} + \text{FN})$$

$$\text{FPR} = \text{FP} / (\text{FP} + \text{TN})$$

threshold	TPR	FPR
<b>0.15</b>	$3/3 = 1$	$3/4 = 0.75$
<b>0.25</b>	$2/3 = 0.67$	$2/4 = 0.5$
<b>0.35</b>	$1/3 = 0.33$	$2/4 = 0.5$
<b>0.55</b>	$1/3 = 0.33$	$1/4 = 0.25$

Area Under the Curve (AUC) sendiri mengukur kemampuan classifier untuk membedakan masing-masing class hasil prediksi.



# Area Under ROC Curve (AUC) for Imbalance Case

## Drawing ROC Curve

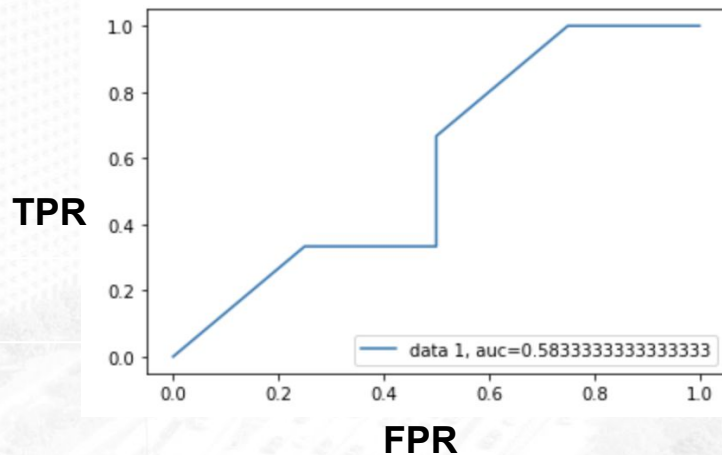
<b>True Label</b>	0	0	1	1	0	0	1
<b>Pred Proba</b>	0.1	0.2	0.2	0.3	0.4	0.7	0.7

ROC curve melakukan plot terhadap True Positive Rate (TPR) dan False Positive Rate (FPR).

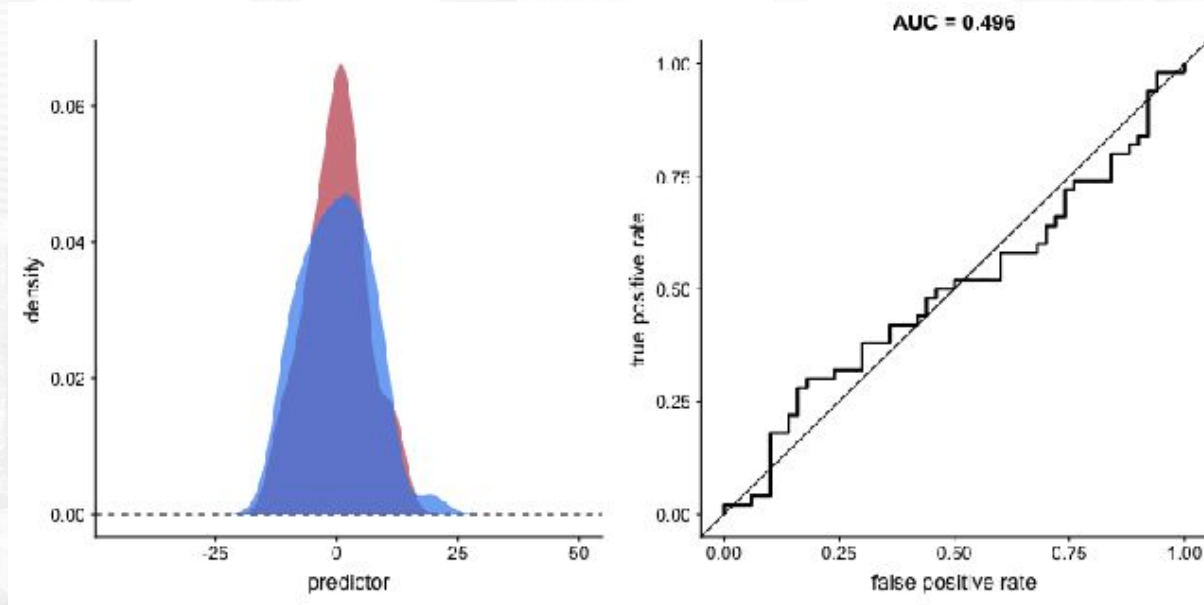
$$\text{TPR} = \text{TP} / (\text{TP} + \text{FN})$$

$$\text{FPR} = \text{FP} / (\text{FP} + \text{TN})$$

threshold	TPR	FPR
<b>0.15</b>	$3/3 = 1$	$3/4 = 0.75$
<b>0.25</b>	$2/3 = 0.67$	$2/4 = 0.5$
<b>0.35</b>	$1/3 = 0.33$	$2/4 = 0.5$
<b>0.55</b>	$1/3 = 0.33$	$1/4 = 0.25$



# Area Under ROC Curve (AUC) for Imbalance Case



**Score AUC yang lebih besar lebih baik,  
dan dapat membedakan prediksi label satu dengan lainnya**

# F1-Score - For Imbalance Class

F1-score menghitung harmonic mean dari precision dan recall

$$F1 = \frac{2 * precision * recall}{precision + recall}$$

Lebih baik digunakan daripada AUC jika kasus imbalance ekstrim. Dan pada kasus tersebut kita butuh fokus untuk prediksi label positif secara tepat.












Contoh: Prediksi kasus penyakit langka

Score F1 yang lebih besar lebih baik.

# Topik Machine Learning Model Evaluation

- ☒ Pengenalan evaluasi model
- ☒ Evaluasi model pada Regresi (Part. 1)
- ☒ Evaluasi model pada Regresi (Part. 2)
- ☒ Implementasi Python - Eval Regresi
- ☒ Evaluasi model pada Klasifikasi (Part. 1)
- ☒ Evaluasi model pada Klasifikasi (Part. 2)
- ☐ Implementasi Python - Eval Klasifikasi
- ☐ Bias-Variance Tradeoff - Overfit & Underfit
- ☐ Bias-Variance Tradeoff - How to solve
- ☐ Model Validation

# Topik Machine Learning Model Evaluation

-  Pengenalan evaluasi model
-  Evaluasi model pada Regresi (Part. 1)
-  Evaluasi model pada Regresi (Part. 2)
-  Implementasi Python - Eval Regresi
-  Evaluasi model pada Klasifikasi (Part. 1)
-  Evaluasi model pada Klasifikasi (Part. 2)
-   Implementasi Python - Eval Klasifikasi
-  Bias-Variance Tradeoff - Overfit & Underfit
-  Bias-Variance Tradeoff - How to solve
-  Model Validation



# Implementation in Python

## With scikit-learn

Scoring	Function
<b>Classification</b>	
'accuracy'	<code>sklearn.metrics.accuracy_score</code>
'average_precision'	<code>sklearn.metrics.average_precision_score</code>
'f1'	<code>sklearn.metrics.f1_score</code>
'precision'	<code>sklearn.metrics.precision_score</code>
'recall'	<code>sklearn.metrics.recall_score</code>
'roc_auc'	<code>sklearn.metrics.roc_auc_score</code>
<b>Clustering</b>	
'adjusted_rand_score'	<code>sklearn.metrics.adjusted_rand_score</code>
<b>Regression</b>	
'mean_absolute_error'	<code>sklearn.metrics.mean_absolute_error</code>
'mean_squared_error'	<code>sklearn.metrics.mean_squared_error</code>
'r2'	<code>sklearn.metrics.r2_score</code>

Contoh: `from sklearn.metrics import accuracy_score`

[https://scikit-learn.org/0.15/modules/model\\_evaluation.html](https://scikit-learn.org/0.15/modules/model_evaluation.html)

# Implementation in Python

## Confusion Matrix (aka. Classification Result)

```
y_true = [1, 0, 1, 0, 0, 1] # actual data
y_pred = [0, 0, 0, 0, 0, 1] # prediksi data

from sklearn.metrics import confusion_matrix # import package dari sklearn
confusion_matrix(y_true, y_pred) # lihat hasil confusion matrix-nya
```

Output:

```
array([[3, 0],
       [2, 1]])
```



		Predicted Label	
		0	1
True Label	0	3 (TN)	0 (FP)
	1	2 (FN)	1 (TP)

[https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion\\_matrix.html#sklearn.metrics.confusion\\_matrix](https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html#sklearn.metrics.confusion_matrix)

# Implementation in Python

## Accuracy, Precision, Recall

```
from sklearn.metrics import accuracy_score  
accuracy_score(y_true, y_pred)
```

Output: 0.6666666666666666

```
from sklearn.metrics import precision_score  
precision_score(y_true, y_pred)
```

```
from sklearn.metrics import recall_score  
recall_score(y_true, y_pred)
```

[https://scikit-learn.org/stable/modules/generated/sklearn.metrics.accuracy\\_score.html](https://scikit-learn.org/stable/modules/generated/sklearn.metrics.accuracy_score.html)

[https://scikit-learn.org/stable/modules/generated/sklearn.metrics.recall\\_score.html#sklearn.metrics.recall\\_score](https://scikit-learn.org/stable/modules/generated/sklearn.metrics.recall_score.html#sklearn.metrics.recall_score)

[https://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision\\_score.html#sklearn.metrics.precision\\_score](https://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision_score.html#sklearn.metrics.precision_score)

# Implementation in Python

## AUC, F1

```
from sklearn.metrics import roc_auc_score  
roc_auc_score(y_true, y_pred)
```

Output: 0.6666666666666666

```
from sklearn.metrics import f1_score  
f1_score(y_true, y_pred)
```

## Challenge Time 2 (2 mins)

```
actual = [1,0,1,1,0,1]  
predicted = [0,0,0,1,1,1]
```

Berapa nilai recall dari hasil prediksi diatas?  
Hitung dengan `recall_score()`



## Challenge Time 2 (2 mins)

```
actual = [1,0,1,1,0,1]  
predicted = [0,0,0,1,1,1]
```












Berapa nilai recall dari hasil prediksi diatas?

0.5

# Topik Machine Learning Model Evaluation

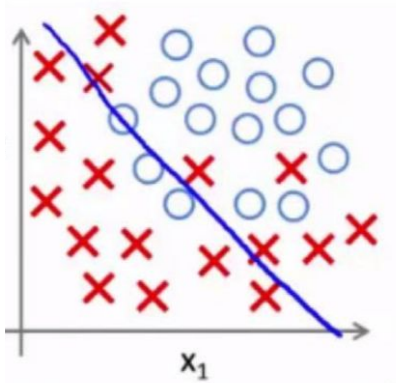
- ☒ Pengenalan evaluasi model
- ☒ Evaluasi model pada Regresi (Part. 1)
- ☒ Evaluasi model pada Regresi (Part. 2)
- ☒ Implementasi Python - Eval Regresi
- ☒ Evaluasi model pada Klasifikasi (Part. 1)
- ☒ Evaluasi model pada Klasifikasi (Part. 2)
- ☒ Implementasi Python - Eval Klasifikasi
- ☐ Bias-Variance Tradeoff - Overfit & Underfit
- ☐ Bias-Variance Tradeoff - How to solve
- ☐ Model Validation

# Topik Machine Learning Model Evaluation

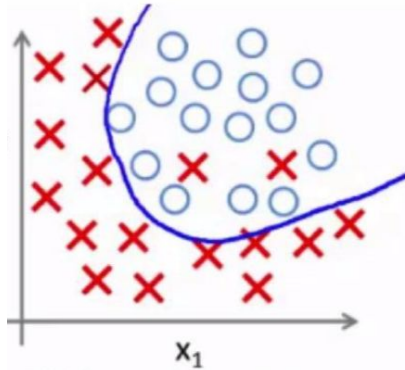
-  Pengenalan evaluasi model
-  Evaluasi model pada Regresi (Part. 1)
-  Evaluasi model pada Regresi (Part. 2)
-  Implementasi Python - Eval Regresi
-  Evaluasi model pada Klasifikasi (Part. 1)
-  Evaluasi model pada Klasifikasi (Part. 2)
-  Implementasi Python - Eval Klasifikasi
-   Bias-Variance Tradeoff - Overfit & Underfit
-  Bias-Variance Tradeoff - How to solve
-  Model Validation

## Challenge Time 3

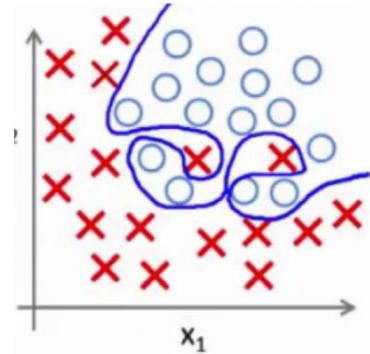
Mana menurut kalian model yang baik? Kenapa?



A



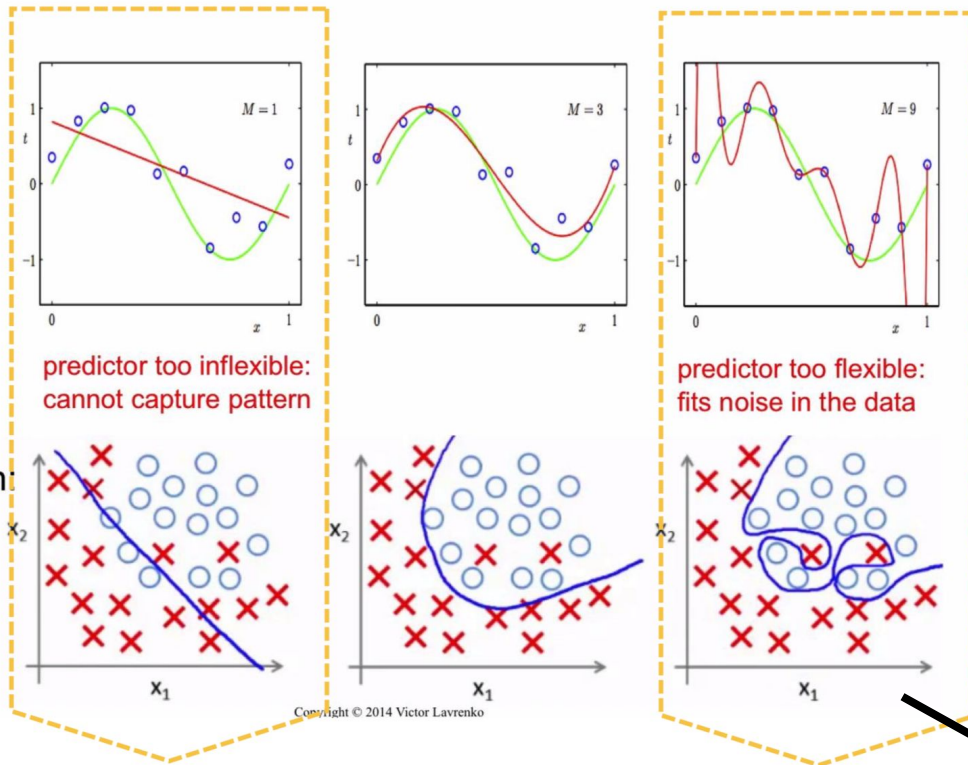
B



C

# Overfitting/Underfitting

Regression:



Classification:

Model yang mempunyai nilai bias yang tinggi jika model tidak dapat menemukan relasi yang tepat antara variabel dan target output (underfit)

Sedangkan jika mempunyai variance yang tinggi jika model modelnya terlalu sensitif terhadap detail-detail kecil pada data, sehingga tidak dapat melakukan prediksi terhadap data yang lebih general



# Bias-Variance Tradeoff

## 1. Bias

Error ketika proses training. Bias yang tinggi -> relasi yang tidak relevan antara feature dan target

## 2. Variance

Error yang melihat seberapa sensitif terhadap model terhadap data training. Variance yang tinggi artinya model banyak menangkap noise, dan jadi tidak general

Bagaimana cara menghitungnya?












```
from mlxtend.evaluate import bias_variance_decomp
```

Referensi: [mlxtend](https://mlxtend.github.io/tutorial_evaluation/bias_variance_decomp.html)

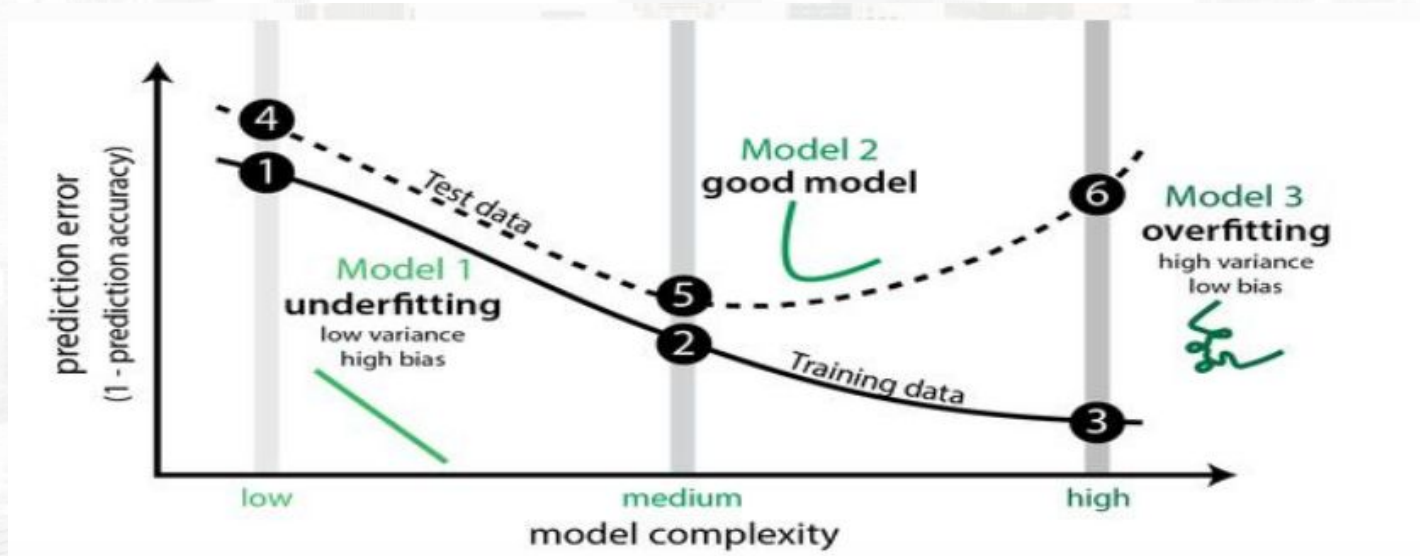
# Topik Machine Learning Model Evaluation

- ☒ Pengenalan evaluasi model
- ☒ Evaluasi model pada Regresi (Part. 1)
- ☒ Evaluasi model pada Regresi (Part. 2)
- ☒ Implementasi Python - Eval Regresi
- ☒ Evaluasi model pada Klasifikasi (Part. 1)
- ☒ Evaluasi model pada Klasifikasi (Part. 2)
- ☒ Implementasi Python - Eval Klasifikasi
- ☒ Bias-Variance Tradeoff - Overfit & Underfit
- ☐ Bias-Variance Tradeoff - How to solve
- ☐ Model Validation

# Topik Machine Learning Model Evaluation

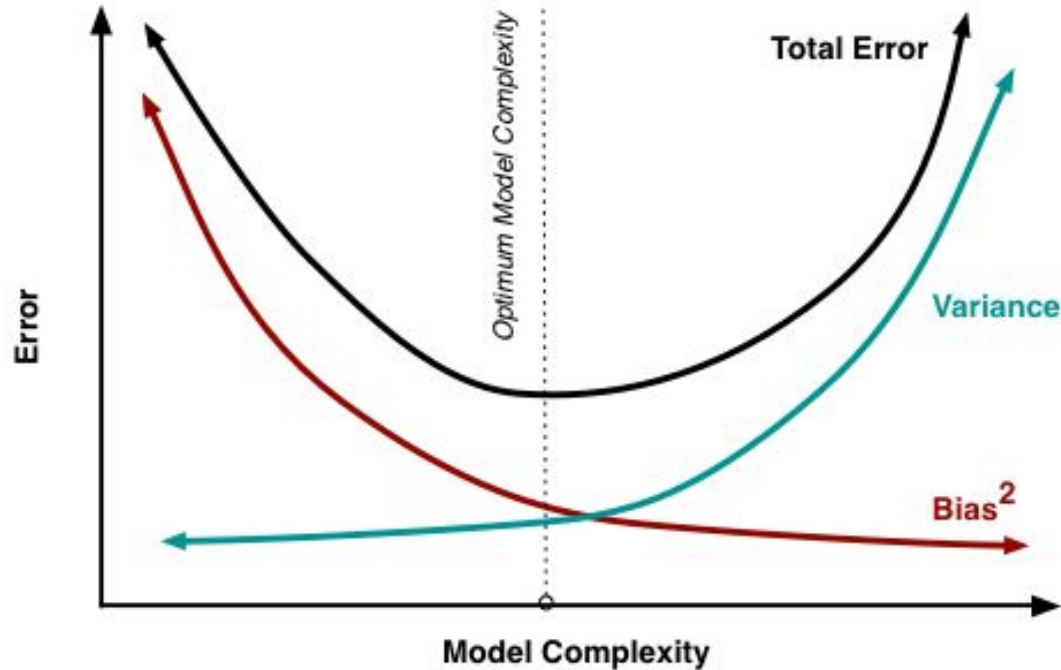
-  Pengenalan evaluasi model
-  Evaluasi model pada Regresi (Part. 1)
-  Evaluasi model pada Regresi (Part. 2)
-  Implementasi Python - Eval Regresi
-  Evaluasi model pada Klasifikasi (Part. 1)
-  Evaluasi model pada Klasifikasi (Part. 2)
-  Implementasi Python - Eval Klasifikasi
-  Bias-Variance Tradeoff - Overfit & Underfit
-   Bias-Variance Tradeoff - How to solve
-  Model Validation

# Model Validation



Semakin kompleks modelnya (dalam artian feature terlalu kompleks, atau terlalu banyak parameter algoritma yang dihitung), semakin besar potensi untuk overfit. Untuk mengurangi overfitting, dikurangi kompleksitasnya dengan feature selection, ataupun algoritma yang lebih sederhana.

# Bias-Variance Trade-off



Demikian juga dengan nilai variance juga akan cenderung lebih tinggi pada model yang lebih kompleks



## Challenge Time 4

Manakah model dan parameter terbaik yang kita pilih?

K	Training score	Testing score
k = 1	1.0	0.727
k = 13	0.782	0.779
k = 200	0.641	0.675

## Challenge Time 4

Manakah model dan parameter terbaik yang kita pilih?

K	Training score	Testing score
k = 1	1.0	0.727
k = 13	0.782	0.779
k = 200	0.641	0.675

<- Overfitting

<- Underfitting

Pilih model yang mempunyai training\_score > testing\_score, namun jaraknya tidak terlalu jauh (masih tolerable)












## Bagaimana Menghindari Overfitting?

- Kurangi kompleksitas model
- Tambah data training
- Lakukan data augmentation  
(menambah data training dengan sedikit modifikasi - SMOTE)
- **Deteksi model validation**
- **Tuning Hyperparameter**

# Topik Machine Learning Model Evaluation

- ☒ Pengenalan evaluasi model
- ☒ Evaluasi model pada Regresi (Part. 1)
- ☒ Evaluasi model pada Regresi (Part. 2)
- ☒ Implementasi Python - Eval Regresi
- ☒ Evaluasi model pada Klasifikasi (Part. 1)
- ☒ Evaluasi model pada Klasifikasi (Part. 2)
- ☒ Implementasi Python - Eval Klasifikasi
- ☒ Bias-Variance Tradeoff - Overfit & Underfit
- ☒ Bias-Variance Tradeoff - How to solve
- ☐ Model Validation

# Topik Machine Learning Model Evaluation

-  Pengenalan evaluasi model
-  Evaluasi model pada Regresi (Part. 1)
-  Evaluasi model pada Regresi (Part. 2)
-  Implementasi Python - Eval Regresi
-  Evaluasi model pada Klasifikasi (Part. 1)
-  Evaluasi model pada Klasifikasi (Part. 2)
-  Implementasi Python - Eval Klasifikasi
-  Bias-Variance Tradeoff - Overfit & Underfit
-  Bias-Variance Tradeoff - How to solve
-   Model Validation

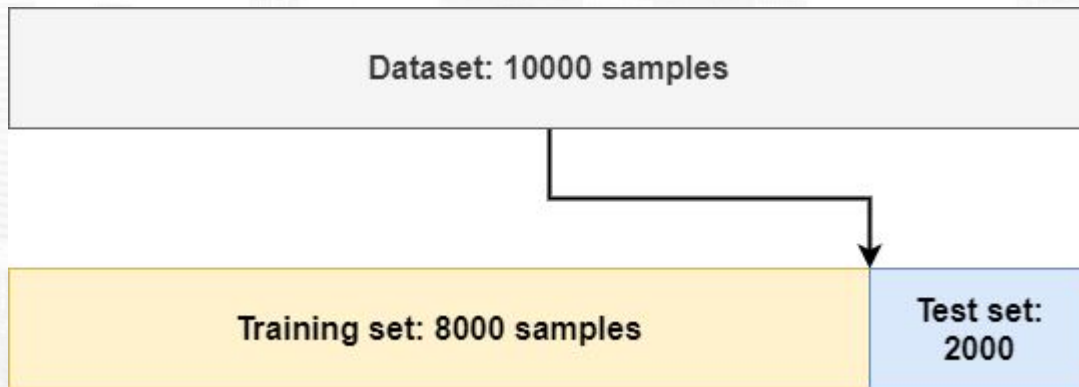


# Model Validation

Pada pertemuan ini kita akan membahas 2 hal:

1. Split Train & Test
2. K-fold Cross Validation

# Split Train & Test



Melakukan pembagian secara random terhadap dataset menjadi:

1. Training set yang digunakan untuk melakukan training pada model dan mendapatkan parameter yang optimal pada model
2. Test set yang digunakan sebagai evaluasi pada model

Rule of thumb: 80:20 atau 70:30

# Split Train & Test di Python

```
from sklearn import model_selection# import package dari sklearn  
  
X_train, X_test, Y_train, Y_test = model_selection.train_test_split(X, Y,  
test_size=test_size)
```

# Split Train & Test di Python

```
from sklearn import model_selection# import package dari sklearn  
  
X_train, X_test, Y_train, Y_test = model_selection.train_test_split(X, Y,  
test_size=test_size)
```

X adalah feature yang kita gunakan  
Y adalah targetnya.

Output dari X\_train dan Y\_train akan digunakan untuk train & fit model  
X\_test dan Y\_test digunakan untuk evaluasi

# Split Train & Test di Python

```
from sklearn import model_selection# import package dari sklearn  
  
X_train, X_test, Y_train, Y_test = model_selection.train_test_split(X, Y,  
test_size=test_size, random_state=42)
```

random\_state digunakan agar split data train dan test yang kita lakukan di-random dengan cara yang sama.



## Membaca feature dan target dari file csv

```
import pandas as pd

dataset = pd.read_csv("kc_house_data.csv")
x = dataset[['sqft_living', 'bedrooms']] # feature yang kita gunakan
y = dataset[['price']] # variable target

#Splitting the data into Train and Test
from sklearn.model_selection import train_test_split
xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size=0.3, random_state=42)
```

kc\_house\_data.csv

price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	...	grade	sqft_above	sqft_basement	yr_built	yr_renovated
221900.0	3	1.00	1180	5650	1.0	0	0	...	7	1180	0	1955	0
538000.0	3	2.25	2570	7242	2.0	0	0	...	7	2170	400	1951	1991
180000.0	2	1.00	770	10000	1.0	0	0	...	6	770	0	1933	0
604000.0	4	3.00	1960	5000	1.0	0	0	...	7	1050	910	1965	0
510000.0	3	2.00	1680	8080	1.0	0	0	...	8	1680	0	1987	0

## Membaca feature dan target dari file csv

```
import pandas as pd

dataset = pd.read_csv("kc_house_data.csv")
x = dataset[['sqft_living', 'bedrooms']] # feature yang kita gunakan
y = dataset[['price']] # variable target

#Splitting the data into Train and Test
from sklearn.model_selection import train_test_split
xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size=0.3, random_state=42)
```

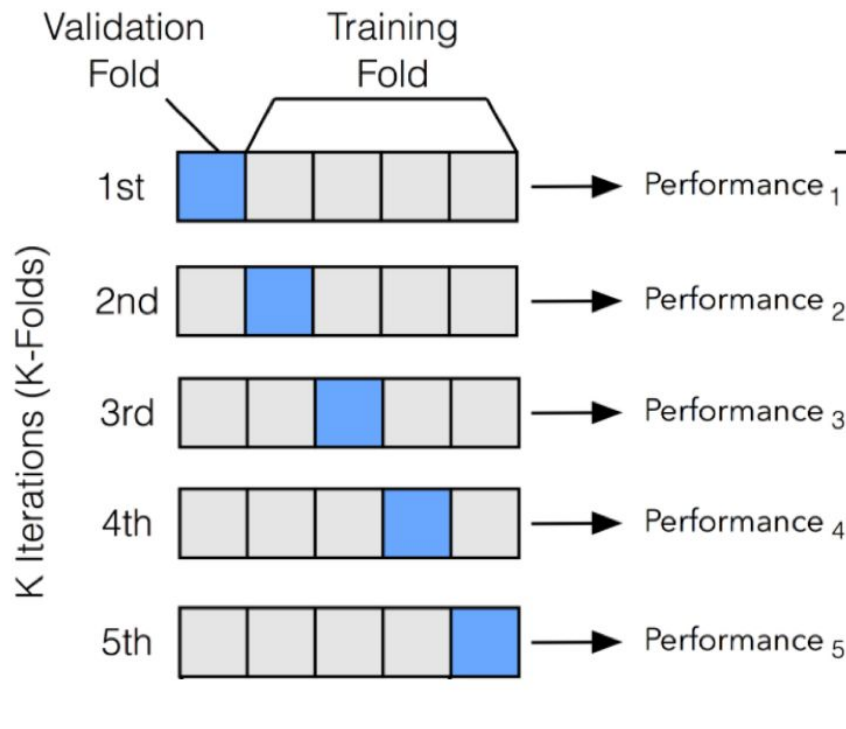
dataset.shape (21613, 21)

xtrain.shape (15129, 2)

xtest.shape (6484, 2)

# K-fold Cross Validation

Cross validation membantu melakukan evaluasi yang lebih menyeluruh dibanding dengan naive split train/test.



Pada k-Fold Cross Validation, melakukan pembagian pada dataset menjadi sebanyak k kelompok (fold), dengan jumlah yang sama, yang masing-masing mempunyai training fold dan validation fold.

Seluruh k-fold akan dihitung performanya satu per satu, dan hasilnya adalah rata-rata dari semuanya.

Performance

$$= \frac{1}{5} \sum_{i=1}^5 \text{Performance}_i$$

# Tips For Cross-Validation

- k-fold cross validation sangat umum digunakan dalam melakukan evaluasi performa machine learning pada data test yang baru
- Nilai k yang biasa digunakan pada k-fold cross validation adalah 3,5,10

# K-fold Cross Validation in Python

```
from sklearn.model_selection import KFold, cross_val_score

model = LinearRegression()
results = cross_val_score(model, X, Y, cv=10, scoring='roc_auc')# calculate score
```

Code ini hanya memperlihatkan contoh bagaimana split train dan test set dengan k-fold cross validation dan menghitungnya dari model.





# Topik Machine Learning Model Evaluation

- ✓ Pengenalan evaluasi model
- ✓ Evaluasi model pada Regresi (Part. 1)
- ✓ Evaluasi model pada Regresi (Part. 2)
- ✓ Implementasi Python - Eval Regresi
- ✓ Evaluasi model pada Klasifikasi (Part. 1)
- ✓ Evaluasi model pada Klasifikasi (Part. 2)
- ✓ Implementasi Python - Eval Klasifikasi
- ✓ Bias-Variance Tradeoff - Overfit & Underfit
- ✓ Bias-Variance Tradeoff - How to solve
- ✓ Model Validation